IN THE CLAIMS

Please amend Claims 1, 14, 15, 19-24, 33 and 35, and cancel Claims 27 and 28 without prejudice, as follows:

1. (Currently amended) A method of debugging a plurality of distributed programs, comprising:

identifying a plurality of processes;

initializing each of said processes;

executing with a single thread of control among said processes; and

continuously switching between said processes to obtain status information relating thereto;

wherein said act of continuously switching comprises switching without determining whether [[an]] a debug event ~~of interest~~ has occurred.

2. (Original) The method of Claim 1, wherein the act of identifying a plurality of processes comprises identifying at least one simulation process and at least one hardware process.

3. (Original) The method of Claim 2, further comprising analyzing said status information to identify at least one or more occurrences or errors within at least one of said distributed programs.

4. (Original) The method of Claim 2, further comprising:

defining at least one object class,

defining at least one first object subclass for said at least one hardware process; and

defining at least one second object subclass for said at least one simulation process.

5. (Previously presented) The method of Claim 2, wherein the act of executing comprises providing at least one first instance variable adapted for control of at least one of said plurality of processes.

6. (Original) The method of Claim 5, further comprising dynamically changing polling times associated with said at least one of said plurality of processes based on the status thereof.

7. (Original) The method of Claim 5, further comprising defining an interface to a first library for said at least one hardware process.

8. (Original) The method of Claim 7, further comprising accessing said first library via said interface in order to provide functions relating to at least one extension instruction.

9. (Original) The method of Claim 8, further comprising defining an interface to a first library for said at least one simulation process.

10. (Original) The method of Claim 8, further comprising accessing said first library via said interface in order to provide functions relating to at least one extension instruction, including the implemention of said at least one extension instruction.

11. (Original) The method of Claim 2, wherein the act of continuously switching comprises cycling between said processes in repeated succession.

12. (Original) The method of Claim 2, wherein the act of initializing comprises:

initializing a first process resident on a first hardware processor; and

initializing a second process on a simulator.

13. (Original) The method of Claim 1, further comprising:

defining a plurality of individual subclasses for each of said plurality of processes; and

implementing at least a portion of said subclasses as a dynamically loadable library.

14. (Currently amended) A system for debugging heterogenous processors, said heterogeneous processors having among them at least two differing instruction sets, comprising:

a first processor having at least one debug process running thereon, and at least one simulation process associated and in data communication with said at least one debug process;

a second processor having a different instruction set then the first, said second processor having said at least one debug process running thereon; and

at least one hardware process in data communication with said at least one debug process;

wherein said at least one simulation and hardware processes are controlled via a single thread.

15. (Currently amended) The system of Claim 14, wherein said first and second processor comprises a digital processor embodied as an integrated circuit, and said at least one debug process comprises a computer program adapted to run on said integrated circuit.

16. (Original) The system of Claim 15, further comprising at least one external port adapted for said data communication with respective ones of said at least one hardware processes.

17. (Original) The system of Claim 16, wherein said at least one simulation process comprises a computer program running on said digital processor.

18. (Original) The system of Claim 17, further comprising a plurality of dynamically loadable libraries, at least a portion of said libraries being adapted for communication with said at least one debug process.

19. (Currently amended) A storage device adapted for use with a computing device, comprising:

a storage medium configured to store a plurality of data thereon; and

a plurality of data stored thereon, said data comprising a computer program adapted to run on a processor, and configured to debug one or more other computer programs using the method comprising:

identifying a plurality of simulator processes;

initializing each of said processes;

executing with a single thread of control among said processes; and

continuously switching between said processes irrespective of the status of a debug event to obtain status information relating thereto.

20. (Currently amended) A method of debugging a plurality of distributed programs across heterogeneous processors, comprising:

identifying a plurality of processes;

defining at least one interface between a debug process and said plurality of processes;

initializing each of said processes;

executing with a single thread of control among said processes; and

selectively permitting at least a portion of said processes to operate while ~~stopping others of said processes~~ simulating the failure of at least one of said heterogeneous processors using a single one of said at least one interface.

21. (Currently amended) A method of optimizing the operation of a multi-processor system comprising a plurality of heterogeneous software processes, comprising:

initializing each of said heterogeneous processes;

executing with a single thread of control among said heterogeneous processes;

iteratively obtaining execution profiling information from at least a portion of said processors; and

optimizing the operation of said system based at least in part on said execution profiling information.

22. (Currently amended) A method of coordinating the operation of at least one simulation process with at least ~~one~~ two heterogeneous hardware processes, comprising:

initializing at least one simulation process;

initializing at least ~~one~~ two heterogeneous hardware ~~process,~~ processes, said hardware processes further comprising at least two different instruction set architectures;

defining at least one interface between a debug process and said plurality of processes;

executing with a single thread of control among said processes; and

controlling said at least one simulation process and said at least ~~one~~ two hardware processes via said at least one interface.

23. (Currently amended) A method of debugging a plurality of extended digital processors using a debug process, said debug process being in data communication with said processors, comprising:

initializing each of said extended digital processors;

executing with a single thread of control among said processors using said debug process;

establishing a minimum polling time for each of said processors; and

obtaining status information from each of said processors based at least in part on said polling interval and outputting said status information to a user perceivable medium, said act of obtaining occurring without determining whether a debug event has occurred.

24. (Currently amended) An apparatus adapted for debug of a plurality of heterogeneous processors, comprising:

a digital processor;

at least one debug process adapted to run on said digital processor; and

a plurality of software processes distributed among said plurality of heterogeneous processors;

wherein said plurality of processes are adapted to gather status information regarding respective ones of said heterogeneous processors as controlled by said debug process, said gathering of status information occurring on a dynamic per-process time interval.

25. (Previously presented) The apparatus of Claim 24, wherein said gathering of status information comprises switching without determining whether or not a debug event has occurred.

26. (Previously presented) The apparatus of Claim 25, wherein at least one of said plurality of software processes comprises a simulator adapted to simulate the operation of an extended processor having at least one extension instruction, said apparatus further comprising an extension library comprising an implementation of said extension instruction.

27. (Cancelled)

28. (Cancelled)

29. (Previously presented) The method of Claim 23, wherein said extended digital processors each comprise at least one extension comprising customized hardware; and

said debug process is adapted to include a processor instance class for at least one of said extended processors, said instance class defining a further interface to dynamically loadable libraries associated with said at least one extension.

30. (Previously presented) The method of Claim 29, wherein said act of obtaining status information comprises obtaining said status information without determining whether a debug event has occurred.

31. (Previously presented) A language-independent method of debugging a plurality of entities selected from the group consisting of: (i) a simulator, and (ii) a digital processor core, the method comprising the following steps performed withion an object-oriented environment:

identifying a plurality of processes associated with said entities;

initializing each of said processes;

executing with a single thread of control among said processes; and

switching on a periodic basis between individual ones of said processes to obtain status information relating thereto irrespective of the status of a debug event associated with one of said processes.

32.(Previously presented) The method of Claim 31, wherein said method further comprises defining an abstract class having a plurality of instance variables, said class providing an interface to at least one of said plurality of processes, said instance variables of said abstract class comprising: (i) variables relating to control of at least one of said entities and examination of its state; or (ii) those relating to synchronization or control of said processes.

33. (Currently amended) A method of debugging a plurality of processes on a plurality of heterogeneous processors using a debugger program, comprising:

initializing a poll delay associated with each process;

determining the state of said debugger program;

if said state comprises a first value, determining the run status of at least one of said processes;

if said at least process is running, determining a process type associated with said at least one process;

if said process type comprises a simulation type, advancing through at least one instruction cycle and checking the status of said simulated process; and

if said process type comprises a hardware type, delaying said debugger program for a predetermined time until a subsequent polling opportunity is available, and determining the status of said hardware processat or after said subsequent polling opportunity[[.]];

wherein said plurality of processes are adapted to gather status information regarding respective ones of said heterogeneous processors as controlled by said debug process, said gathering of status information occurring on a dynamic per-process time interval.

34. (Previously presented) The method of Claim 33, further comprising:

utilizing said debugger program to check a value of simulator variable to determine whether any currently running ones of said processes are executing in a simulator; and if so, returning said debugger program immediately to said act of determining the state so that said process(es) executing in a simulator will run as quickly as possible.

35. (Currently amended) A method of debugging a plurality of heterogenous processes, said plurality of processes comprising a plurality of simulators and at least hardware processor, the method comprising:

running said at least one hardware processor so as to execute at substantially the speed with

which execution would occur if it were not running under the control of a debugger;

monitoring the status of said at least one hardware processor using a single thread of control, said single thread not significantly slowing said execution; and

running said plurality of simulators so as to maintain synchronization therebetween;

wherein said acts of running and monitoring <u>said heterogeneous processes</u> are all <u>performed</u> ~~perfromed~~ using a single debugging program.